Software Requirements

Descriptions and specifications of a system

What is a requirement?

- May range from
  - a high-level abstract statement of a service or
  - a statement of a system constraint to a detailed mathematical functional specification
- Requirements may be used for
  - a bid for a contract
    - must be open to interpretation
  - the basis for the contract itself
    - must be defined in detail
- Both the above statements may be called requirements

Example

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function	Add node
Description	Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.
Inputs	Node type, Node position, Design identifier.
Source	Node type and Node position are input by the user, Design identifier from the database.
Outputs	Design identifier.
Destination	The design database. The design is committed to the database on completion of the operation.
Requires	Design graph rooted at input design identifier.
Pre-condition	The design is open and displayed on the user's screen.
Post-condition	The design is unchanged apart from the addition of a node of the specified type at the given position.
Side-effects	None

Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1

Example

……

4.A.5 The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name.

……

Types of requirements

- Written for customers
  - User requirements
    - Statements in natural language plus diagrams of the services the system provides and its operational constraints.
- Written as a contract between client and contractor
  - System requirements
    - A structured document setting out detailed descriptions of the system services.
- Written for developers
  - Software specification
    - A detailed software description which can serve as a basis for a design or implementation.

User requirements readers

- Client managers
- System end-users
- Client engineers
- Contractor managers
- System architects

System requirements readers

- System end-users
- Client engineers
- System architects
- Software developers

Software specification readers

- Client engineers (maybe)
- System architects
- Software developers

We will come back to user and system requirements

Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

Functional requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail

Examples of functional requirements

1. The user shall be able to search either all of the initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in the document store.
3. Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Requirements imprecision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users
- Consider the term 'appropriate viewers'
  - User intention - special purpose viewer for each different document type
  - Developer interpretation - Provide a text viewer that shows the contents of the document

Requirements completeness and consistency

- In principle, requirements should be both complete and consistent
- Complete
  - They should include descriptions of all facilities required
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities
- In practice, it is difficult (?impossible?) to produce a complete and consistent requirements document

What requirements are these?

- It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set
- The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95
- The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

Non-functional requirements

- constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

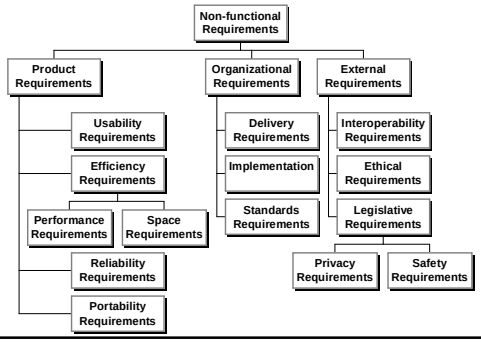
Non-functional requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular system, programming language or development method
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organizational requirements
  - Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirement types



Non-functional requirements examples

- Product requirement
  - 4.C.8 It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set
- Organizational requirement
  - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95
- External requirement
  - 7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

Goals and requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested
- Goals are helpful to developers as they convey the intentions of the system users

Examples

- A system goal
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.
- A verifiable non-functional requirement
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Requirements measures

Property	Measure
Speed	·Processed transactions/second ·User/event response time ·Screen refresh time
Size	·K Bytes ·Number of RAM chips
Ease of use	·Training time ·Number of help frames
Reliability	·Mean time to failure ·Probability of unavailability ·Rate of failure occurrence ·Availability
Robustness	·Time to restart after failure ·Percentage of events causing failure ·Probability of data corruption on failure
Portability	·Percentage of target dependent statements ·Number of target systems

Requirements interaction

- Conflicts between different non-functional requirements are common in complex systems
- Spacecraft system
  - To minimize weight, the number of separate chips in the system should be minimized
  - To minimize power consumption, lower power chips should be used
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

Domain requirements

- Requirements that come from the application domain of the system and that reflect characteristics of that domain

Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

Library system domain requirements

- There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
- Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

Domain requirements problems

- Understandability
  - Requirements are expressed in the language of the application domain
  - This is often not understood by software engineers developing the system
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit

Back to user and system requirements

User requirements

- Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge
- User requirements are defined using natural language, tables and diagrams

Database requirement

……
4.A.5 The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name.
……

Requirement problems

- Database requirements includes both conceptual and detailed information
  - Describes the concept of configuration control facilities
  - Includes the detail that objects may be accessed using an incomplete name

Editor grid requirement

……
2.6 Grid facilities To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimetres at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.
……

Requirement problems

- Grid requirement mixes three different kinds of requirement
  - Conceptual functional requirement (the need for a grid)
  - Non-functional requirement (grid units)
  - Non-functional UI requirement (grid switching)

Why the problems?

Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up
- Requirements mix-up
  - Several different requirements may be expressed together

Structured presentation

2.6 Grid facilities

2.6.1 **The editor shall provide a grid facility where a matrix of horizontal and vertical lines provide a background to the editor window.** This grid shall be a passive grid where the alignment of entities is the user's responsibility.

Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities 'snap-to' grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

Detailed user requirement

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their design.**

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.

2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.

3. The user should then drag the node symbol to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

Guidelines for writing requirements

- Invent a standard format and use it for all requirements
- Use language in a consistent way. Use "shall" for mandatory requirements, "should" for desirable requirements
- Use text highlighting to identify key parts of the requirement
- Avoid the use of computer jargon

System requirements

- More detailed specifications of user requirements
- Serve as a basis for designing the system
- May be used as part of the system contract

Problems with NL specification

- Ambiguity
  - The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult
- Over-flexibility
  - The same thing may be said in a number of different ways in the specification
- Lack of modularisation
  - NL structures are inadequate to structure system requirements

Alternatives to NL specification

	Notation	Description
	Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
	Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system.
	Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT (Ross, 1977; Schoman and Ross, 1977). More recently, use-case descriptions (Jacobsen, Christerson et al., 1993) have been used. I discuss these in the following chapter.
	Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured language specifications

- A limited form of natural language may be used to express requirements
- This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification
- Often best supported using a forms-based approach

Form-based specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any)

Form-based node specification

ECLIPSE/Workstation/Tools/DE/FS/3.5.1	
Function	Add node
Description	Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.
Inputs	Node type, Node position, Design identifier.
Source	Node type and Node position are input by the user, Design identifier from the database.
Outputs	Design identifier.
Destination	The design database. The design is committed to the database on completion of the operation.
Requires	Design graph rooted at input design identifier.
Pre-condition	The design is open and displayed on the user's screen.
Post-condition	The design is unchanged apart from the addition of a node of the specified type at the given position.
Side-effects	None
Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1	

PDL-based requirements definition

- Requirements may be defined using a language like a programming language but with more flexibility of expression
- Most appropriate in two situations
  - Where an operation is specified as a sequence of actions and the order is important
  - When hardware and software interfaces have to be specified
- Disadvantages are
  - The program definition language (PDL) may not be sufficiently expressive to define domain concepts
  - The specification will be taken as a design rather than a specification

Part of an ATM specification

```
class ATM {
    // declarations here
    public static void main (String args[]) throws InvalidCard {
        try {
            thisCard.read () ;  // may throw InvalidCard exception
            pin = KeyPad.readPin () ; attempts = 1 ;
            while ( !thisCard.pin.equals (pin) & attempts < 4 )
                {       pin = KeyPad.readPin () ;  attempts = attempts + 1 ;
                }
                if (!thisCard.pin.equals (pin))
                    throw new InvalidCard ("Bad PIN");
            thisBalance = thisCard.getBalance () ;
            do {  Screen.prompt (" Please select a service ") ;
                service = Screen.touchKey () ;
                switch (service) {
                    case Services.withdrawalWithReceipt:
                                receiptRequired = true ;
```

PDL disadvantages

- PDL may not be sufficiently expressive to express the system functionality in an understandable way
- Notation is only understandable to people with programming language knowledge
- The requirement may be taken as a design specification rather than a model to help understand the system

Interface specification

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements
- Three types of interface may have to be defined
  - Procedural interfaces
  - Data structures that are exchanged
  - Data representations
- Formal notations are an effective technique for interface specification

PDL interface description

```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

        void initialize ( Printer p ) ;
        void print ( Printer p, PrintDoc d ) ;
        void displayPrintQueue ( Printer p ) ;
        void cancelPrintJob (Printer p, PrintDoc d) ;
        void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

Viewpoint-oriented elicitation

- Stakeholders represent different ways of looking at a problem or problem viewpoints
- This multi-perspective analysis is important as there is no single correct way to analyze system requirements

Banking ATM system

- The example used here is an auto-teller system which provides some automated banking services
- I use a very simplified system which offers some services to customers of the bank who own the system and a narrower range of services to other customers
- Services include cash withdrawal, message passing (send a message to request a service), ordering a statement and transferring funds

Autoteller viewpoints

- Bank customers
- Representatives of other banks
- Hardware and software maintenance engineers
- Marketing department
- Bank managers and counter staff
- Database administrators and security staff
- Communications engineers
- Personnel department

Types of viewpoints

- Data sources or sinks
  - Viewpoints are responsible for producing or consuming data.
  - Analysis involves checking that data is produced and consumed and that assumptions about the source and sink of data are valid
- Representation frameworks
  - Viewpoints represent particular types of system model.
  - These may be compared to discover requirements that would be missed using a single representation. Particularly suitable for real-time systems
- Receivers of services
  - Viewpoints are external to the system and receive services from it.
  - Most suited to interactive systems

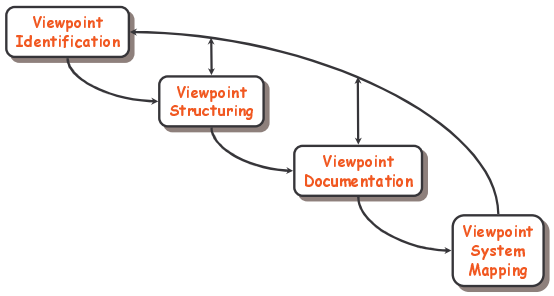
External viewpoints

- Natural to think of end-users as receivers of system services
- Viewpoints are a natural way to structure requirements elicitation
- It is relatively easy to decide if a viewpoint is valid
- Viewpoints and services may be used to structure non-functional requirements

Method-based analysis

- Widely used approach to requirements analysis. Depends on the application of a structured method to understand the system
- Methods have different emphases. Some are designed for requirements elicitation, others are close to design methods
- A viewpoint-oriented method (VORD) is used as an example here. It also illustrates the use of viewpoints

The VORD method



VORD process model

- Viewpoint identification
  - Discover viewpoints which receive system services and identify the services provided to each viewpoint
- Viewpoint structuring
  - Group related viewpoints into a hierarchy. Common services are provided at higher-levels in the hierarchy
- Viewpoint documentation
  - Refine the description of the identified viewpoints and services
- Viewpoint-system mapping
  - Transform the analysis to an object-oriented design

VORD standard forms

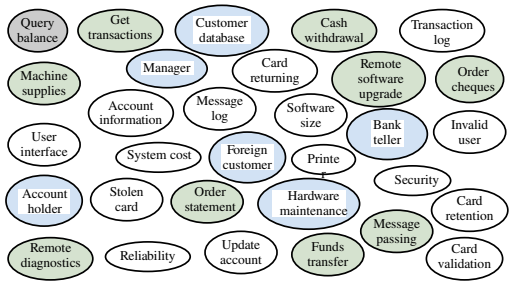
Viewpoint template

Reference:	The viewpoint name.
Attributes:	Attributes providing viewpoint information.
Events:	A reference to a set of event scenarios describing how the system reacts to viewpoint events.
Services	A reference to a set of service descriptions.
Sub-VPs:	The names of sub-viewpoints.

Service template

Reference:	The service name.
Rationale:	Reason why the service is provided.
Specification:	Reference to a list of service specifications. These may be expressed in different notations.
Viewpoints:	List of viewpoint names receiving the service.
Non-functional requirements:	Reference to a set of non-functional requirements which constrain the service.
Provider:	Reference to a list of system objects which provide the service.

Viewpoint identification



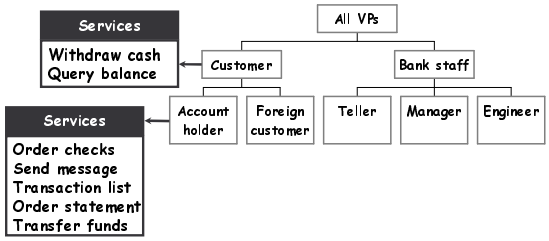
Viewpoint service information

Account Holder	Foreign Customer	Bank Teller
Service List	**Service List**	**Service List**
1. Withdraw cash 2. Query balance 3. Order checks 4. Send message 5. Transaction list 6. Order statement 7. Transfer funds	1. Withdraw cash 2. Query balance	1. Run diagnostics 2. Add cash 3. Add paper 4. Send Message

Viewpoint data/control

Account Holder	Control Input	Data Input
	1. Start transaction 2. Cancel transaction 3. End transaction 4. Select service	1. Card details 2. PIN 3. Amount required 4. Message

Viewpoint hierarchy



Customer/cash withdrawal templates

- Reference
  - Customer
- Attributes
  - Account number
  - PIN
  - Start transaction
- Events
  - Select service
  - Cancel transaction
  - End transaction
- Services
  - Cash withdrawal
  - Balance enquiry
- Sub-VPs
  - Account holder
  - Foreign customer

- Reference
  - Cash withdrawal
- Rationale
  - To improve customer service and reduce paperwork
- Specification
  - Users choose this service by pressing the cash withdrawal button. They then enter the amount required. This is confirmed and, if the funds are low, the balance is delivered
- VPs
  - Customer
- Non-functional requirements
  - Deliver cash within 1 minute of amount being confirmed
- Provider
  - *Filled in later*

Scenarios

- Scenarios are descriptions of how a system is used in practice
- They are helpful in requirements elicitation as people can relate to these more readily than abstract statement of what they require from a system
- Scenarios are particularly useful for adding detail to an outline requirements description

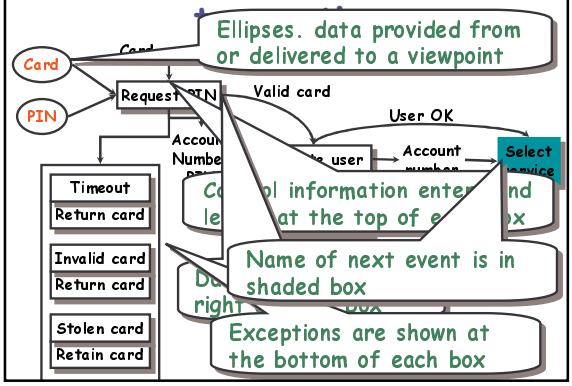
Scenario descriptions

- System state at the beginning of the scenario
- Normal flow of events in the scenario
- What can go wrong and how this is handled
- Other concurrent activities
- System state on completion of the scenario

Event scenarios

- Event scenarios may be used to describe how a system responds to the occurrence of some particular event such as 'start transaction'
- VORD includes a diagrammatic convention for event scenarios.
  - Data provided and delivered
  - Control information
  - Exception processing
  - The next expected event

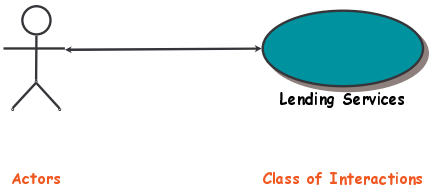
Event scenario - start



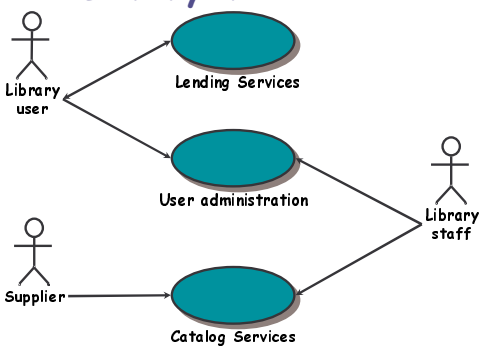
Use cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself
- A set of use cases should describe all possible interactions with the system

Lending use-case



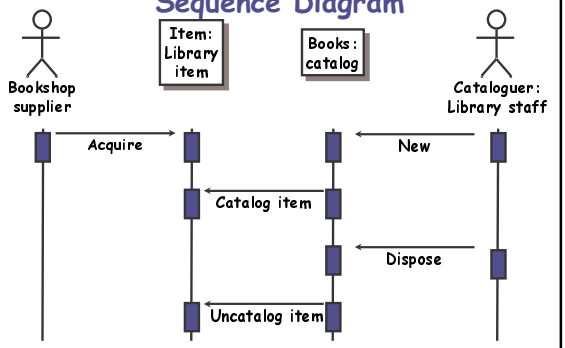
Library use-cases



Sequence Diagrams

- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system

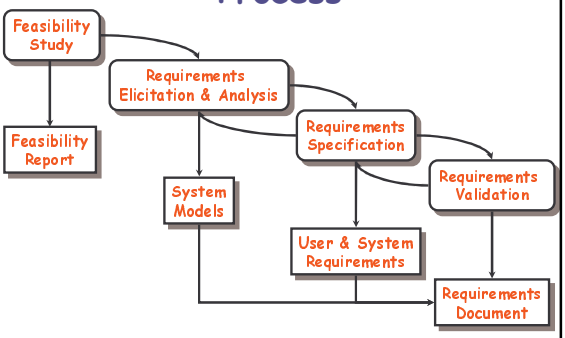
Catalogue management: Sequence Diagram



Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organization developing the requirements
- However, there are a number of generic activities common to all processes
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management

The Requirements Engineering Process



Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile
- A short focused study that checks
  - If the system contributes to organizational objectives
  - If the system can be engineered using current technology and within budget
  - If the system can be integrated with other systems that are used

Feasibility study implementation

- Based on information assessment (what is required), information collection and report writing
- Questions for people in the organization
  - What if the system wasn't implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?

Elicit: by Webster dictionary

Main Entry: **elic·it**
Pronunciation: i-'li-s&t
Function: *transitive verb*
Etymology: Latin elicitus, past participle of elicere, from e- + lacere to allure
Date: 1605
1 : to draw forth or bring out (something latent or potential) <*hypnotism elicited his hidden fears***>**
2 : to call forth or draw out (as information or a response) <*her remarks elicited cheers***>**

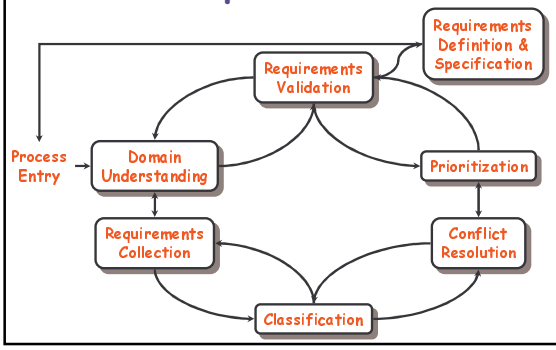
Elicitation

- Sometimes called requirements elicitation or requirements discovery
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders

Requirements Analysis

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organizational and political factors may influence the system requirements
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change

The requirements analysis process



Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error

Requirements Validation

- Validity. Does the system provide the functions that best support the customer's needs?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included?
- Realism. Can the requirements be implemented given available budget and technology
- Verifiability. Can the requirements be checked?

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability
- Automated consistency analysis
  - Checking the consistency of a structured requirements description

Requirements reviews

- Regular reviews should be held while the requirements definition is being formulated
- Both client and contractor staff should be involved in reviews
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage

Review checks

- Verifiability. Is the requirement realistically testable?
- Comprehensibility. Is the requirement properly understood?
- Traceability. Is the origin of the requirement clearly stated?
- Adaptability. Can the requirement be changed without a large impact on other requirements?

Requirements management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development
- Requirements are inevitably incomplete and inconsistent
  - New requirements emerge during the process as business needs change and a better understanding of the system is developed
  - Different viewpoints have different requirements and these are often contradictory

Requirements management planning

- During the requirements engineering process, you have to plan:
  - Requirements identification
    - How requirements are individually identified
  - A change management process
    - The process followed when analyzing a requirements change
  - Traceability policies
    - The amount of information about requirements relationships that is maintained
  - CASE tool support
    - The tool support required to help manage requirements change

Traceability

- Traceability is concerned with the relationships between requirements, their sources and the system design
- Source traceability
  - Links from requirements to stakeholders who proposed these requirements
- Requirements traceability
  - Links between dependent requirements
- Design traceability
  - Links from the requirements to the design

A traceability matrix

Req id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1			R					
1.2		U	U			R		U
1.3	R			R				
2.1			R		U			U
2.2								U
2.3		R		U				
3.1								R
3.2							R	

U = "uses the requirement", R = "Some other weaker relationship"

CASE tool support

- Requirements storage
  - Requirements should be managed in a secure, managed data store
- Change management
  - The process of change management is a workflow process whose stages can be defined and information flow between these stages partially automated
- Traceability management
  - Automated retrieval of the links between requirements

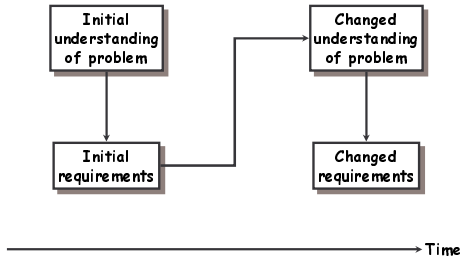
Enduring and volatile requirements

- Enduring requirements. Stable requirements derived from the core activity of the customer organisation. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- Volatile requirements. Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Requirements change

- The priority of requirements from different viewpoints changes during the development process
- System customers may specify requirements from a business perspective that conflict with end-user requirements
- The business and technical environment of the system changes during its development

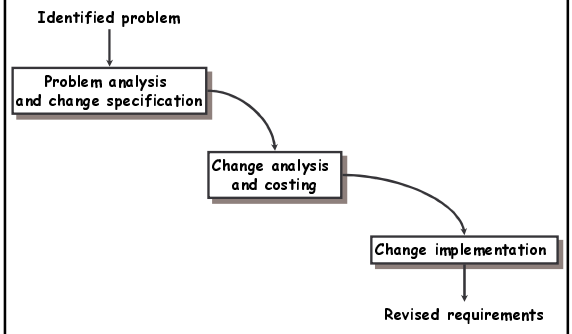
Requirements evolution



Requirements change management

- Should apply to all proposed changes to the requirements
- Principal stages
  - Problem analysis. Discuss requirements problem and propose change
  - Change analysis and costing. Assess effects of change on other requirements
  - Change implementation. Modify requirements document and other documents to reflect change

Requirements change management



The requirements document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Users of a requirements document

- System customers
  - Specify the requirements and read them to check that they meet their needs
- Managers
  - Use the requirements document to plan a bid for the system and to plan the system
- System engineers
  - Use the requirements to understand what system is to be developed
- System test engineers
  - Use the requirements to develop validation tests for the system
- System maintenance engineers
  - Use the requirements to help understand the system and the relationship between its parts

Requirements document requirements

- Specify external system behaviour
- Specify implementation constraints
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system i.e. predict changes
- Characterise responses to unexpected events

IEEE requirements standard

- Introduction
- General description
- Specific requirements
- Appendices
- Index
- This is a generic structure that must be instantiated for specific systems